Programmation 2 Spring, 2026

TD 7: reductions

1. On suppose qu'on dispose d'une fonction `sommepaire` qui, étant donné une liste `A` et une cible `x`, retourne `True` s'il existe deux éléments dans `A` dont la somme égale `x`, `False` sinon.

   À l'aide de cette fonction, écrivez une fonction `sommetriple` qui reçoit une liste `A` et une cible `x`, et retourne `True` s'il existe trois éléments dans `A` dont la somme égale `x`, `False` sinon.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple:` pour `A = [1, 2, 7, 11, 15]` et `x = 18`, la fonction renvoie `True` ($1 + 2 + 15 = 18$).
   pour `A = [1, 2, 7, 11, 15]` et `x = 17`, la fonction renvoie `False`.

2. On suppose qu'on dispose d'une fonction `bin(n).count('1')` qui retourne le nombre de bits valant 1 dans sa représentation binaire de `n`. À l'aide de cette fonction, écrivez une fonction `est_puissance_de_deux` qui reçoit un entier positif `n` et retourne `True` si `n` est une puissance de deux, `False` sinon.

   `Exemple:` pour `n = 8`, la fonction renvoie `True`; pour `n = 6`, elle renvoie `False`.

3. On suppose qu'on dispose d'une fonction `compterinversions` qui, étant donné une liste `A` d'entiers *distincte*, retourne le nombre de paires `(i, j)` avec $0 \leq$ `i` $<$ `j` $\leq$ `len(A)-1` et `A[i] > A[j]`. À l'aide de cette fonction, écrivez une fonction `esttriee` qui reçoit une liste `A` et retourne `True` si et seulement si `A` est triée dans l'ordre **décroissant**.

   Vous n'avez pas le droit d'utiliser des boucles.

   `Exemple:` pour `A = [4, 3, 2, 1]`, la fonction renvoie `True`.
   pour `A = [4, 3, 1, 2]`, la fonction renvoie `False`.

4. On suppose qu'on dispose d'une fonction `commonprefix` qui, étant donné *une liste* `A` contenant deux chaînes (donc `A = [s1, s2]`), retourne leur plus long préfixe commun. À l'aide de cette fonction, écrivez une fonction `min_chaines` qui reçoit une liste `A` de chaînes et retourne la chaîne minimale dans `A` (par ordre lexicographique).

   Vous n'avez pas le droit de comparer deux chaînes de caractères, mais vous pouvez comparer deux caractères.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple:` pour `A = ["aeb", "abf", "abd", "ac"]`, la fonction renvoie `"abd"`.

5. On vous donne une liste `A` contenant n nombres.

   Écrivez une fonction `findTwoContiguous` qui prend `A` et renvoie la distance entre les deux nombres les plus proches de `A`. Vous pouvez supposer que vous avez accès à une fonction `sorted(A)` qui renverra une liste triée par ordre croissant.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple:` pour `A = [ 10, 78, 102, 20, 44, 69 ]`, la fonction renvoie `9`.

6. Écrivez une fonction `findMajority` qui prend une liste d'entiers `A` et renvoie l'élément qui apparaît plus de `len(A) / 2` fois dans `A`. Vous pouvez supposer qu'il existe un tel élément dans `A`. Vous pouvez supposer que vous avez accès à une fonction `sorted(A)` qui renverra une liste triée par ordre croissant.

Vous n'avez pas le droit d'utiliser des boucles.

7. On vous donne une liste `A` contenant n nombres.

   Écrivez une fonction `findClosest` qui prend `A`, un nombre x et un entier k. Elle renvoie les k nombres de `A` les plus proches de x.

   Supposons que vous ayez accès à une fonction `sorted(B, key=lambda element: element[0])` qui prend une liste de listes `B` et retourne une nouvelle liste triée selon l'ordre croissant du premier élément (d'indice 0) de chaque sous-liste.

   Par exemple : `sortedB = sorted([[3, 'a'], [1, 'b'], [4, 'c']], key=lambda element: element[0])` va retourner `[[1, 'b'], [3, 'a'], [4, 'c']]`.

   Vous n'avez pas le droit d'utiliser des boucles imbriquées.

   `Exemple` : pour `A = [4, 1, 3, 8, 10]`, `x = 5` et `k = 3`, la fonction renvoie `[4, 3, 8]` (les 3 nombres les plus proches de 5 sont 4, 3 et 8, dans n'importe quel ordre).

8. (**difficile**) On suppose qu'on dispose d'une fonction `to_binary_string` qui, étant donné un entier positif x, renvoie une string représentant x en binaire.

   À l'aide de cette fonction, écrivez une fonction `allsubsets` qui reçoit une liste `A` et retourne une liste de listes correspondant à tous les sous-ensembles de `A`.

   `Example:` pour `A = ['a', 'af', 41.34]`, la fonction renvoie

   `[[], ['a'], ['af'], ['a', 'af'], [41.34], ['a', 41.34], ['af', 41.34], ['a', 'af', 41.34]]`.